The shift minimisation personnel task scheduling problem: a new hybrid approach and computational insights

Pieter Smet Tony Wauters Mihail Mihaylov
Greet Vanden Berghe

KU Leuven, Department of Computer Science, CODeS, iMinds
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
{pieter.smet, tony.wauters, mihail.mihaylov, greet.vandenberghe }@cs.kuleuven.be

January 28, 2013
revised October 30, 2013

Abstract

Assigning scheduled tasks to a multi-skilled workforce is a known NP-complete problem with many applications in health care, services, logistics and manufacturing. Optimising the use and composition of costly and scarce resources such as staff has major implications on any organisation's health. The present paper introduces a new, versatile two-phase matheuristic approach to the shift minimisation personnel task scheduling problem, which considers assigning tasks to a set of multi-skilled employees, whose working times have been determined beforehand. Computational results show that the new hybrid method is capable of finding, for the first time, optimal solutions for all benchmark instances from the literature, in very limited computation time. The influence of a set of problem instance features on the performance of different algorithms is investigated in order to discover what makes particular problem instances harder than others. These insights are useful when deciding on organisational policies to better manage various operational aspects related to workforce. The empirical hardness

results enable to generate hard problem instances. A set of new challenging instances is now available to the academic community.

Keywords: Heuristics · Mathematical programming · Scheduling · Timetabling

1 Introduction and related work

After several years of research, the wide domain of personnel rostering remains a relevant academic subject (Van den Bergh et al., 2013). As personnel costs have become the major part of operational expenses, it is ever so important to organise a given workforce as efficiently as possible in order to reduce the associated costs and to increase employee satisfaction (Wright and Mahar, 2013).

In the personnel scheduling literature, assigning shifts to personnel is often the most fine-grained level at which the allocation is being discussed (Ernst et al., 2004; Ormeci et al., 2014; Maenhout and Vanhoucke, 2013b). The assignment of particular tasks to employees during a shift is often not incorporated in the construction of rosters. In some cases, employees know beforehand which tasks to perform during their working hours. In hospitals, for example, nurses know exactly what they must do in each shift (Burke et al., 2004; Li et al., 2012). Indeed, some tasks such as meal distribution and hygienic or medical care of patients need to be conducted within set time frames. In many other cases however, tasks are assigned to employees in an ad hoc manner, often resulting in excess resource utilisation. It is therefore recommendable to incorporate task assignment in the construction of rosters for employees, in order to reduce operational expenses while still maintaining a high quality of service. Maenhout and Vanhoucke (2013a) point out the importance of such an integrative approach to achieve a more efficient and effective allocation in a different context. While solving the integrated problem does lead to better overall results, it raises the complexity significantly. Addressing such a problem in several stages makes it thus easier to solve to a satisfactory level.

The present paper follows the latter reasoning and discusses a heuristic approach to the problem of assigning tasks to multi-skilled employees, while minimising the number of employees. This problem was introduced by Krishnamoorthy and Ernst (2001) as the shift minimisation personnel task

scheduling problem (SMPTSP). Krishnamoorthy et al. (2012) introduced a benchmark dataset for the SMPTSP. Using a Lagrangean relaxation based approach that combines two problem specific heuristics, they were able to find 135 feasible solutions and 67 optimal solutions out of the 137 benchmark instances. Furthermore, Krishnamoorthy et al. (2012) discussed some interesting properties of the problem and presented algorithms for solving particular subproblems of the SMPTSP. Smet and Vanden Berghe (2012) applied a hybrid local search using a *fix and optimise* strategy to the dataset and thereby found 68 new best solutions and 81 optimal solutions. They also provided new lower bounds for 43 instances.

The SMPTSP is similar to the interval scheduling problem (Kolen et al., 2007), in which a set of jobs with fixed start and end times are given, as well as a set of machines on which the jobs should be processed. The goal is to decide which jobs to process and on which machines, while e.g. maximising profit associated with the jobs. The SMPTSP differs from the basic interval scheduling problem in that it requires all jobs (tasks) to be assigned while not all machines (employees) can process each job.

Musliu et al. (2004) discuss the minimum shift design problem (MSD) in which the goal is to find the minimum number of shifts and the number of employees to be assigned to these shifts, such that the deviation from a given set of coverage requirements is minimised. In essence, this problem deals with a similar challenge as the SMPTSP, i.e. to decide on a minimal workforce for carrying out all the work. However, the MSD does not incorporate the assignment of tasks within the shifts, thus omitting a level of detail addressed in the SMPTSP.

The present paper introduces a two-phase approach that can be classified as a matheuristic, as it combines the strengths of both heuristics and exact approaches (Maniezzo et al., 2009). This family of hybrid approaches recently gained a lot of attention because of their ability to solve problems for which traditional (meta)heuristics or exact approaches fail to find good quality solutions. Della Croce and Salassa (2012) describe a VNS-based matheuristic for a real world nurse rostering problem in which different neigbourhoods are searched by including additional constraints into the model. These constraints fix particular heuristically selected variables. Computational experiments show that this matheuristic approach significantly outperforms exact commercial general purpose solvers. Matheuristic approaches have been applied to many other hard combinatorial optimisation problems including vehicle routing (Doerner and Schmid, 2010), permutation flow shop schedul-

ing (Della Croce et al., 2011) and the multidimensional knapsack problem (Hanafi et al., 2010).

The rest of the paper is organised as follows. First, the contributions and their practical relevance are outlined in Section 2. The problem definition is provided in Section 3. Section 4 presents different constructive heuristics for the SMPTSP, as well as a heuristic improvement procedure based on local branching. A discussion on the algorithmic design is included in Section 5. Furthermore, the performance of the presented algorithm is compared to that of recent approaches from the literature. The influence of instance specific characteristics on algorithmic performance is discussed in Section 6. Based on these empirical hardness results, new hard instances are introduced in Section 7. Finally, Section 8 concludes the paper and presents future research directions.

2 Contributions and practical relevance

The first major contribution of the paper is the introduction of a hybrid heuristic approach, which, at present, improves upon all state of the art algorithms for the SMPTSP. A study comparing the performance against recently published solution techniques from the literature, and against a commercial solver demonstrates the efficiency and effectiveness of the new hybrid heuristic. Furthermore, it is shown that the algorithm is able to find, for the first time, an optimal solution for all instances from the only publicly available dataset at present (Krishnamoorthy et al., 2012). The second contribution is an investigation of SMPTSP properties that affect algorithmic performance. Finally, as a third contribution, based on the empirical hardness study, a new benchmark dataset is generated that presents more challenging problem instances than the existing ones proposed by Krishnamoorthy et al. (2012).

The efficient allocation of scarce resources is an ever-present issue for management, particularly when these resources cause high expenses for the organisation. This is especially true for the SMPTSP since inefficient assignment of the available workforce can lead to significant costs, for example when hiring additional temporary workers becomes inevitable. Manual planners often simplify the assignment by making abstraction of intricate problem properties such as start and end times of tasks or qualification requirements. However, many organisations require this complexity to be incorporated in the decision making process. Ignoring it would render any decision support

approach inapplicable. Efficient algorithms for problems entailing the full complexity enable better decision making on both strategic and operational level. The former is achieved by determining the optimal composition of an organisation's workforce, and the latter by efficiently deploying these costly resources and thereby reducing operational expenses.

Insights into organisational characteristics that influence algorithmic performance allow policies for managing crucial aspects of an organisation to be (re)defined in order to significantly improve an algorithm's efficiency, resulting in an effective decision support system. By establishing such an understanding for two properties of the SMPTSP, better methodologies for defining tasks and organising the staff skill mix can be conceived to improve overall efficiency and relieve some of the pressure on a flexible workforce.

3 Problem definition

Let $J = \{1, ..., n\}$ be the set of tasks to be assigned to employees and $W = \{1, ..., m\}$ the set of employees. Each task $j \in J$ has a duration d_j, a start time s_j and a finish time $f_j = s_j + d_j$. Each employee w has a set of tasks $T_w \subseteq J$ that she/he can perform. Similarly, for each task j, a set $P_j \subseteq W$ exists, which contains all employees that can perform j. Both T_w and P_j are defined based on qualifications, time windows of tasks and availabilities of employees.

An interval graph $G = (J, A)$ can be defined with J the set of nodes and A the set of arcs. Two nodes i and j are connected when their respective time intervals, $[s_i,f_i]$ and $[s_j,f_j]$, overlap. The set of maximal cliques in the interval graph is defined as C. This set can be found in polynomial time by first sorting the nodes based on start time and then applying a forward pass algorithm. The set $C = \{K_1, ..., K_t\}$ consists of sets $K_t \subseteq J$ such that any pair of tasks in K_t overlap in time and K_t is maximal. No tasks in $J \setminus K_t$ overlap with any of the tasks in K_t. In terms of the SMPTSP, it is obvious that overlapping tasks, represented by nodes in K_t, should be assigned to different employees. For each employee w, the set of maximal cliques $C^w = \{K_1^w, ..., K_t^w\}$ is constructed in the same way as C, while only the set of tasks for which the employee is qualified is considered. An employee w can only be assigned to one of the tasks from each set $K_t^w \in C^w$. This prevents overlapping assignments in a solution.

To solve the SMPTSP, a feasible solution has to be found in which all

tasks in J are assigned to qualified employees from W in a non-preemptive manner, while minimising the total number of employees.

Two sets of decision variables are defined:

$$x_{jw} = \begin{cases} 1 & \text{if task } j \in J \text{ is assigned to employee } w \in W \\ 0 & \text{otherwise} \end{cases}$$

$$y_w = \begin{cases} 1 & \text{if employee } w \in W \text{ is active, meaning that } w \text{ has at least one task} \\ 0 & \text{otherwise} \end{cases}$$

The SMPTSP can now be defined as follows (Krishnamoorthy et al., 2012):

$$min \sum_{w \in W} y_w \tag{1}$$

$$s.t. \sum_{w \in P_j} x_{jw} = 1 \qquad \forall\ j \in J \tag{2}$$

$$\sum_{j \in K_t^w} x_{jw} \leq y_w \qquad \forall\ w \in W,\ K_t^w \in C^w \tag{3}$$

$$0 \leq y_w \leq 1 \qquad \forall\ w \in W \tag{4}$$

$$x_{jw} \in \{0, 1\} \qquad \forall\ j \in J,\ w \in W \tag{5}$$

The objective function (1) states that the number of employees should be minimised. Constraints (2) ensure that each task is performed by exactly one employee, and that no infeasible assignments in terms of qualifications are made. Constraints (3) make sure that tasks are only assigned to active employees and that tasks assigned to an employee do not overlap. Finally, constraints (4) and (5) set bounds for the decision variables.

The SMPTSP can be seen as an application of list colouring to interval graphs, which is NP-complete (Bonomo et al., 2006). Colours correspond to employees and vertices correspond to tasks. Two vertices are connected whenever the corresponding tasks overlap in time. The qualifications of the employees are represented by the list of feasible colours on each vertex. Other applications of list colouring to interval graphs include classroom allocation (Carter and Tovey, 1992) and register assignment (Zeitlhofer and Wess, 2003).

4 Solution procedure

We present a two-phase hybrid heuristic algorithm based on the principles of matheuristics. A constructive heuristic first generates an initial solution, which is improved in the second phase. Section 4.1 describes several constructive heuristics for providing an initial solution. The improvement heuristic is presented in Section 4.2.

4.1 Constructive heuristics

Three different constructive approaches are presented: *first fit*, *best fit* and a *constructive matheuristic* algorithm.

4.1.1 First fit and best fit heuristics

Krishnamoorthy et al. (2012) state that when the qualification constraints of the SMPTSP are relaxed, i.e. when all employees are qualified to perform all tasks, the resulting problem can be solved in polynomial time with a forward pass maximal clique algorithm on an interval graph (Gupta et al., 1979). This algorithm assigns all tasks in order of increasing starting time, considering first, if possible, an employee who already has tasks assigned. This property is incorporated in the *first fit* and *best fit* constructive heuristics for the SMPTSP. In both heuristics all tasks are first ordered by start time in ascending order. Ties are broken by taking into account the qualifications of employees. For this purpose, the tasks are additionally ordered by the number of qualified personnel able to perform them, also in ascending order. This results in an ordering in which tasks with the smallest number of feasible personnel appear before the other ones. These highly constrained tasks, which are the most difficult to assign, are thus first to be assigned.

An additional mechanism is introduced to ensure that the first fit and best fit constructive heuristics find feasible solutions in cases where tasks can only be performed by a limited number of employees. Whenever a task j cannot be assigned to a feasible employee due to other overlapping assignments, a qualified employee is randomly selected and her/his assigned tasks overlapping with j are removed. Task j is then assigned to this employee and the removed tasks are reassigned to other employees later on in the procedure.

The aforementioned steps outline a general framework for both the first fit and best fit constructive heuristics. The first fit constructive heuristic

assigns tasks to the first feasible employee (Algorithm 1).

Algorithm 1 First fit constructive heuristic

Input: $J :=$ Tasks to be assigned
$P_j :=$ Employees qualified for task $j \in J$
$s_j :=$ Start time of task $j \in J$
$R_w := \emptyset$ {tasks assigned to employee w}
Output: A (partial) solution for the SMPTSP

```
1: Order all j ∈ J by (s_j + |P_j|) in ascending order
2: while J ≠ ∅ do
3:    Remove task j from the first position in J
4:    Assign j to the first feasible employee
5:    if Cannot feasibly assign j to any qualified employee then
6:       Select random employee w ∈ P_j
7:       O_j := Tasks overlapping with task j
8:       Remove all tasks in O_j from R_w
9:       Assign j to employee w
10:      Add tasks in O_j to the list of tasks to be assigned J
11:   end if
12: end while
13: return R_w, ∀w {the assigned tasks for each employee}
```

The best fit constructive heuristic is designed by modifying line 4 in Algorithm 1 such that it assigns the tasks to the *best* feasible employee instead of to the *first* feasible employee. The best employee is identified by the largest sum of assigned task durations $\sum_{j \in R_w} d_j$, with R_w the set of currently assigned tasks to employee w. This way, employees' schedules will include as many tasks as possible, thereby minimising the number of employees.

It is worth noting that both the first fit and best fit constructive heuristics do not guarantee to be finite, i.e. it is possible that they enter an infinite loop in which two sets of tasks are repeatedly assigned to and deassigned from the same set of employees. In order to ensure finite behaviour of these constructive heuristics, a termination criterion is to be included, which stops the algorithm whenever it enters an infinite loop. As a result, tasks may remain unassigned after the constructive heuristics have finished.

4.1.2 Constructive matheuristic

In addition to the first fit and best fit constructive heuristics, we also introduce a constructive matheuristic (CMH), which uses a mathematical solver

to construct a solution by optimally solving subproblems one by one. The subproblems are defined by selecting a subset of $b \leq m$ employees $W' \subseteq W$. These subproblems are sequentially solved using a mathematical solver. Instead of minimising the number of employees, the objective is to maximise the sum of assigned task durations in order to implicitly reduce the number of employees:

$$max \sum_{w \in W'} \sum_{j \in R_w} d_j \tag{6}$$

with R_w the set of tasks assigned to employee w. Only the last subproblem is reoptimised for a second time with the original objective to minimise the number of employees.

Algorithm 2 shows the pseudocode of the constructive matheuristic.

Algorithm 2 Constructive matheuristic

Input: J := set of tasks to be assigned
W := set of employees
b := number of employees in one subproblem
Output: A (partial) solution for the SMPTSP
1: **while** $W \neq \emptyset$ **do**
2: W' := sample and remove b employees from W {delineate the subproblem}
3: Solve the subproblem for W'
4: Remove the tasks assigned to W' from J
5: **if** $J = \emptyset$ **then**
6: Reoptimise W' using the original objective (Equation (1))
7: **end if**
8: **end while**
9: **return** $R_w, \forall w$ {the assigned tasks for each employee}

4.1.3 Infeasibility issues

Initial experiments showed that, particularly for small problem instances with $m < 100$, the constructive heuristics are not always able to assign all tasks. The result can thus be infeasible. This problem is coped with in the second phase of the presented approach, which is a local branching based improvement heuristic. This (general) improvement heuristic is particular in that it does not require a feasible initial solution. Infeasible starting solutions are repaired during the execution of the improvement heuristic.

4.2 Improvement heuristic

After generating the initial solution with one of the aforementioned constructive heuristics, an improvement procedure attempts to further reduce the number of employees, given that the initial solution was not yet optimal. For this purpose, we present a matheuristic based on the idea of local branching (Fischetti and Lodi, 2003).

A solution x is constructed, in which at most k binary variables have flipped values with respect to a given reference solution $\bar{x}$. This is enforced by adding the asymmetric *Hamming distance constraint* (7) to the original mathematical model.

$$\sum_{j \in J} \sum_{w \in W} \bar{x}_{jw}(1 - x_{jw}) \leq k'(= k/2) \tag{7}$$

In the context of the SMPTSP, k' corresponds to the maximum number of tasks that can be (re)assigned, given the reference solution $\bar{x}$. In our improvement heuristic, Constraint (7) is added to the mathematical model and solved to optimality using a general purpose mathematical solver. If the new solution x is not better than the given solution $\bar{x}$, or if the solution's objective value is equal to the lower bound, the procedure stops, else x is set as the new reference solution and the previous steps are reiterated.

Algorithm 3 describes this improvement procedure.

Algorithm 3 clearly builds upon the matheuristic concept of combining integer programming and heuristic search by embedding an optimal approach to solving subproblems in an iterative improvement framework. Due to the absence of problem specific elements, Algorithm 3 also presents a versatile improvement heuristic, applicable to a large class of problems. It can thus be seen as a general method for which only the development of a mathematical model is required. Moreover, in contrast to many other general improvement algorithms, it does not require a feasible starting solution. In some occasions, it is worthwhile to spend effort in designing a constructive heuristic to provide a good initial solution to yield better results. Feasible, or almost feasible, starting solutions will furthermore benefit the algorithm's performance since this will prevent steps 4 and 5 from being executed excessively.

Algorithm 3 Local branching improvement heuristic

Input: $F(x)$ evaluation function of x
LB := lower bound on the evaluation function value
$\bar{x}$:= initial reference solution
k' := maximum number of tasks to reassign
Output: A solution for the SMPTSP

```
improved := true
while improved and F(x̄) ≠ LB and stop criterion not met do
    x := solve the model with the Hamming distance constraint with k' given x̄
    if x is not feasible then
        k' := k' + 1
    else if F(x) < F(x̄) then
        x̄ := x
    else
        improved := false
    end if
end while
return x̄
```

5 Computational results

The behaviour of the proposed heuristics is evaluated by analysing the results of a series of experiments. Furthermore, the quality of the new approaches is compared with the best known results from the literature.

5.1 Experimental setup

All instances from the benchmark dataset from Krishnamoorthy et al. (2012)[1] are used for the experiments and evaluation. The dimensions of the instances range from small (23 employees and 40 tasks) to very large (245 employees and 2105 tasks). On average, all employees can perform either 33% or 66% of the tasks.

All experiments are carried out on an Intel Core i7-2600 at 3.4GHz with 8GB RAM operating on Windows 7. All algorithms are coded in Java. Gurobi 5.1.0 is used as general purpose mathematical solver. Each run is repeated ten times with computation time limited to 1800 seconds per run.

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ptaskinfo.html`

5.2 New lower bounds

In order to facilitate evaluation of algorithmic performance, we present a new lower bound, which improves on the best reported lower bounds from the literature (Smet and Vanden Berghe, 2012).

Proposition 1 *The size of the largest clique from the set C is a valid lower bound for the SMPTSP.*

This is a trivial lower bound since it corresponds to the minimum number of employees needed to cover the largest number of overlapping tasks. Section 3 describes a polynomial time algorithm to calculate this lower bound. This bound does not provide the minimum number of employees required in a solution for the SMPTSP, since it does not take into account the employees' qualifications. However, the computational experiments show that the thus obtained lower bounds are equal to the optimal solution quality for the entire benchmark dataset provided by Krishnamoorthy et al. (2012).

5.3 Constructive heuristics

We compare the performance of the first fit, best fit and CMH with two different block sizes ($b = 10$ and $b = 15$). These values were determined after preliminary experiments. Table 1 summarises the performance of the different constructive algorithms for all 137 available benchmark instances. The lower bound from Section 5.2 was used for determining whether a solution is optimal. Detailed computational results can be found on a dedicated web page [2].

	First fit	**Best fit**	**CMH b = 10**	**CMH b = 15**
Number of optimal solutions	13	22	118	131
Average solution quality	125.35	125.78	123.01	122.61
Average calculation time (seconds)	0.02	0.11	12.26	44.93
Maximum calculation time (seconds)	0.21	1.10	169.30	889.80

Table 1: Summary of results for the constructive heuristics.

Best fit generates more optimal solutions than first fit, whereas, first fit has a slightly better average solution quality, meaning that for instances that cannot be solved optimally with best fit, first fit obtains better solutions. The calculation times required for both best fit and first fit is negligible.

[2] `http://allserv.kahosl.be/~pieter/smptsp.html`

The constructive matheuristic produces significantly more optimal solutions. This improved solution quality comes at the cost of an increased calculation time on large instances. The time required by CMH depends on the block size, which also influences the quality of the constructed solution. Figure 1 illustrates this trade-off for an instance with 211 employees and 1647 tasks. When the block size increases, the subproblems become larger and thus require more calculation time. However, larger block sizes mean that higher numbers of employees are considered in each subproblem, which can result in better solutions.

Based on the results from Table 1, a block size of $b = 10$ was used in future experiments to generate the initial solution for the improvement heuristic.

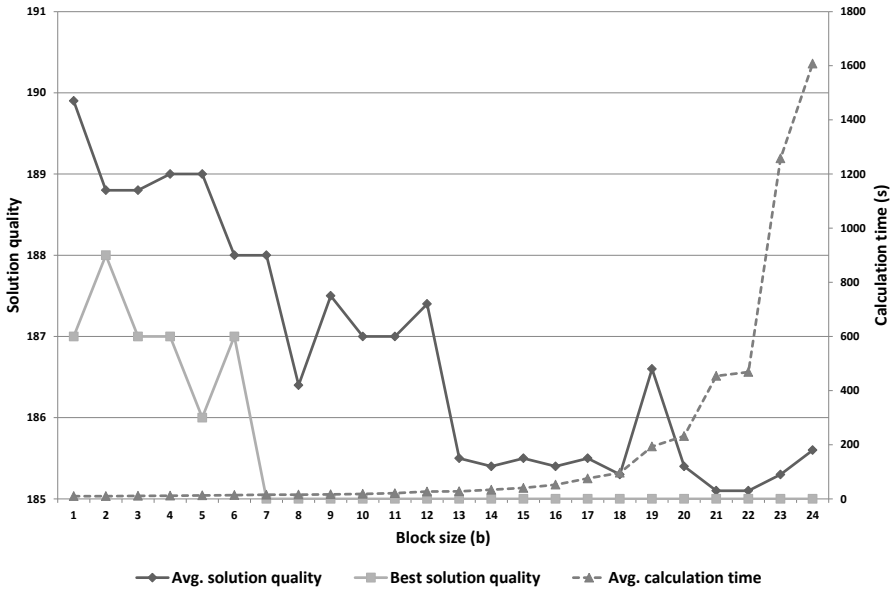

Figure 1: Average and best solution quality and average calculation time for the CMH with varying block size $b = [1, 24]$ on instance 133_211_1647_33.

This decomposition procedure may affect the quality of solutions. In order to determine the influence of how employees are aggregated into subproblems, an experiment was set up in which three approaches were compared. First, employees were selected for subproblems based on the number of tasks they can perform, in ascending order, such that the first subproblem contains the most restricted employees in terms of number of tasks they can perform.

Second, this order was reversed, so that the most qualified employees are selected first. Finally, employees were randomly selected for each subproblem. Table 2 shows the average performance of these three approaches on 50 instances with number of employees ranging from 88 to 415 and number of tasks from 777 to 2105. These results indicate that the employee selection for the subproblem does not influence the performance of the constructive matheuristic. It is important to note, however, that this result is obtained based on instances with a random skilling structure. The presence of a structure in the employee skilling may lead to different conclusions, nevertheless, the random selection provides a certain degree of robustness, which is desirable when the skilling structure varies among departments in the same organisation or when the staff skill mix does not exhibit a clear structure.

	Ascending order	**Descending order**	**Random order**
Average solution quality	160.22	160.32	160.31
Average calculation time	22.19	23.82	23.79
Number of feasible solutions	50	50	50

Table 2: Impact of employee selection for the subproblems.

5.4 Improvement heuristic

Instances for which the constructive matheuristic (CMH $b = 10$) did not produce an optimal solution are further optimised by the local branching based improvement heuristic (LBIH). The parameter k' is set as a function of the problem size and is calculated as $k' = f\sqrt{n}$. The rationale behind this being that, for instances with many tasks, the number of tasks allowed to be reassigned should not be too large since this would make the calculation time for solving the Hamming distance model unacceptable. A linear relation between k' and the number of tasks n would thus not be suitable, therefore the square root of n was chosen, multiplied by a factor f. For a larger f, more tasks will be allowed to move when solving the model with the Hamming distance constraint. Figure 2 shows, for f varying between 0.5 and 5, the best and average solution quality and average solution time of ten runs, for all instances from the benchmark dataset. The results indicate that when more tasks are allowed to move, better solutions can be obtained, but at the cost of an increased calculation time. Based on these experiments, $f = 4.5$ proved to be the most appropriate choice.

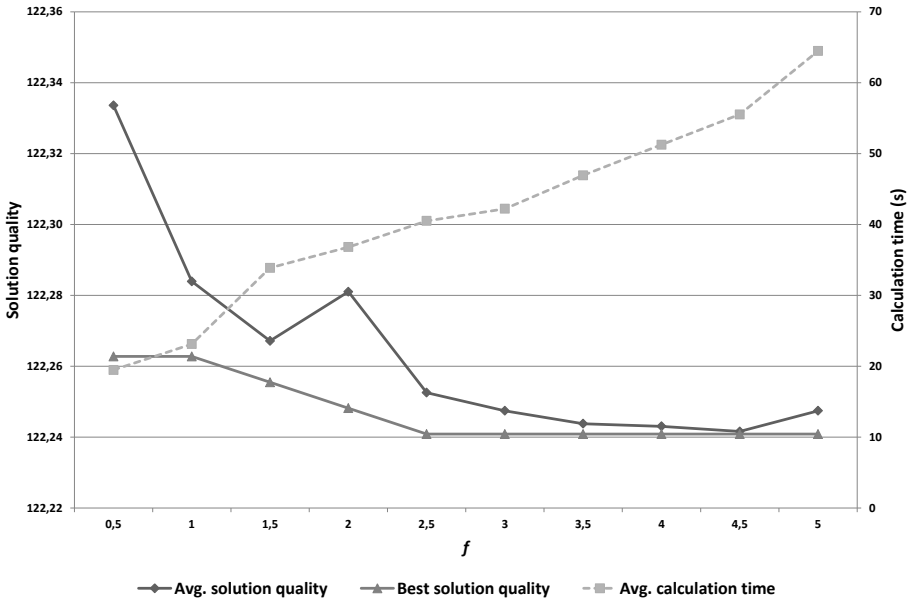

Figure 2: Average and best solution quality and average calculation time for the LBIH with varying parameter $f = [0.5, 5]$ on all instances.

The combined matheuristic CMH+LBIH ($b = 10$, $f = 4.5$) was compared with a commercial general purpose mathematical solver (Gurobi 5.1.0) and with two methods recently reported in the literature: a Lagrangean relaxation based heuristic from Krishnamoorthy et al. (2012) (KEB12) and a fix and optimise heuristic from Smet and Vanden Berghe (2012) (SV12). Table 3 presents the summarised results. A dedicated web page [3] provides the detailed computational results. The reported calculation times are total times, i.e. the sum of computation time of the constructive heuristic and the time required by the improvement heuristic.

	MIP	**KEB12**	**SV12**	**CMH+LBIH (b = 10, f = 4.5)**
Number of optimal solutions	88	67	81	137
Average solution quality	129.88	127.00	123.04	122.24
Average calculation time (seconds)	-	-	958.91	55.52
Maximum calculation time (seconds)	1800.0	1800.00	1800.00	604.87

Table 3: Summary of results for different approaches for the SMPTSP.

[3] `http://allserv.kahosl.be/~pieter/smptsp.html`

The CMH+LBIH ($b = 10$, $f = 4.5$) approach finds, for the first time, an optimal solution for all the 137 instances, requiring much less than the allowed calculation time. On average, the presented method requires 55 seconds to find an optimal solution while the worst case still only takes little more than ten minutes to produce an optimal solution. Compared to the other approaches, CMH+LBIH ($b = 10$, $f = 4.5$) thus performs significantly better, both in terms of solution quality and required calculation time.

6 Empirical hardness

In order to understand the behaviour of algorithms for the SMPTSP, we conducted a series of experiments to determine what makes particular instances easy or hard for different algorithms. It is generally known that, while the complexity of a problem can be established as hard, easy instances of that problem may exist (Leyton-Brown et al., 2002). Identification of relevant hardness features enables development of powerful portfolio techniques (Messelis and De Causmaecker, 2014).

After performing an initial statistical analysis of the performance of different algorithms on the dataset from Krishnamoorthy et al. (2012), the *multi-skilling level* and average task duration showed to be most influential to the hardness of the problem. In this section, their influence on the algorithmic performance for the SMPTSP is investigated.

6.1 Multi-skilling level

The multi-skilling level of an instance is the percentage of the total number of tasks each employee is qualified for, on average. When this level is 100%, each employee can perform all tasks. We set up a series of experiments to determine the influence of this instance feature on the performance of both heuristics and on a general purpose mathematical solver. An instance generator, developed according to the description of Krishnamoorthy et al. (2012), was used to generate a set of new instances in which the multi-skilling level varied from 5% to 100%. For each level, ten random instances were constructed. The reported results are the average (or median in case of computation time) of one run on each of the ten instances.

Figure 3 shows that the gap to the lower bound for the three constructive heuristics, presented in Section 4.1, decreases when the multi-skilling level

increases. Recall from Section 4.1 that the constructive heuristics use ideas from the forward pass maximal clique algorithm for interval graphs. When the multi-skilling level is 100%, the SMPTSP reduces to exactly that problem, thus making it easier for the constructive heuristics to find good solutions. For instances with a multi-skilling level lower than 15%, the heuristics were unable to find a feasible solution.

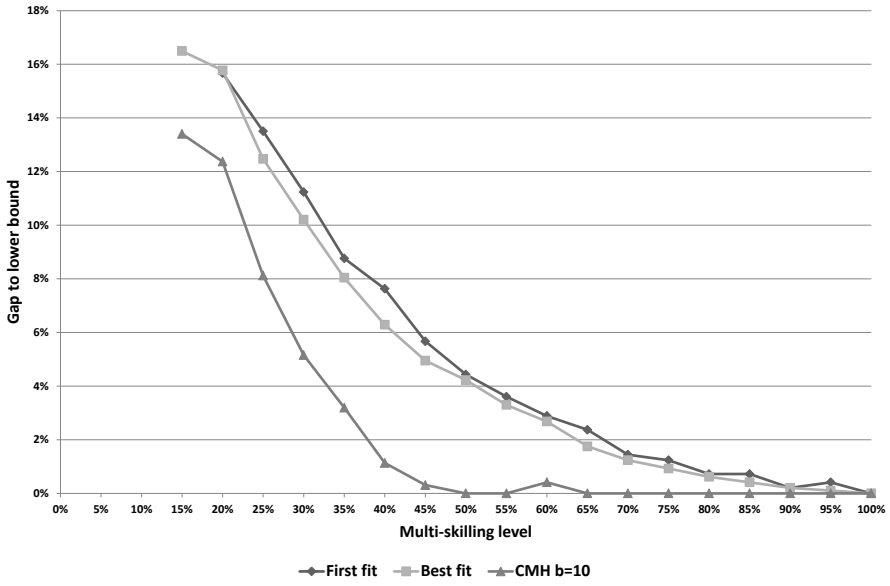

Figure 3: Average gap from lower bound for the constructive heuristics with varying multi-skilling level (113 employees, 1112 tasks, 90% tightness).

We conducted an analogous experiment in which the standard mathematical model was solved with a commercial general purpose solver (Gurobi 5.1.0), using the computation time required for finding an optimal solution as a measurement for hardness. Figure 4 shows, for different configurations of the solver, that it takes longer to find an optimal solution when the multi-skilling level increases. The peak in computation time at around 35% is particular, since this was also one of the two multi-skilling levels used in the original dataset, thus making those instances notably hard for a general purpose solver or any algorithm using it as a subroutine.

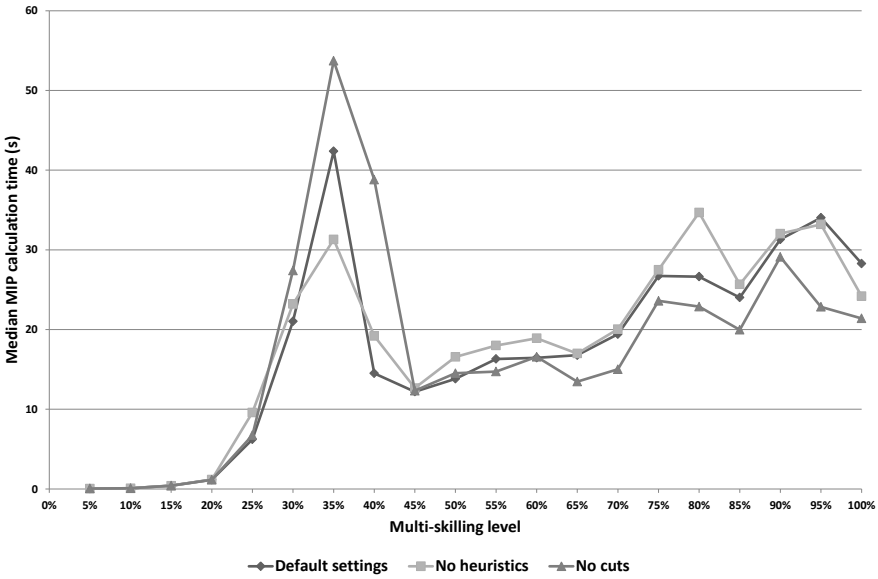

Figure 4: Median computation time in seconds for different configurations with varying multi-skilling level (33 employees, 337 tasks, 90% tightness).

6.2 Average task duration

A similar series of experiments was conducted to investigate the effect of the average task duration on algorithmic performance. The experimental setup was the same as described in Section 6.1, but the distribution from which task lengths are sampled was varied, while keeping the number of employees, number of tasks and multi-skilling level constant. As in the original dataset, the task lengths were sampled from a triangular distribution $Tri(\alpha, \beta, \gamma)$, with β varying between 100 and 440. The ratios of α and γ to β are kept constant: $\alpha = \beta - 100$ and $\gamma = \beta + 100$.

Figure 5 shows that, for all three constructive heuristics, the average gap to the lower bound decreases when tasks become longer. This is particularly clear for the first fit and best fit constructive heuristics. The constructive matheuristic also exhibits this behaviour.

Figure 6 shows the same trend for different configurations of a general purpose solver: longer tasks make instances easier to solve. Note that the number of employees and tasks is kept constant, and therefore the number of variables in the model also remains the same, such that this factor does

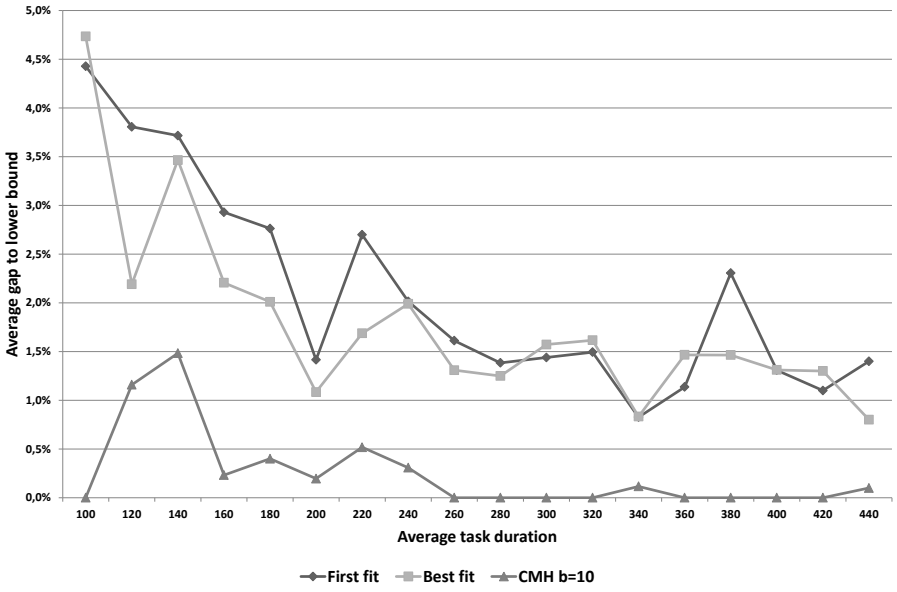

Figure 5: Average gap from lower bound for the constructive heuristics with varying average task duration (100 employees, 300 tasks, 60% skilling).

not alter the solver's performance.

These results show that suitable policies for defining the tasks and staff skill mix in an organisation will strongly influence an algorithm's effectiveness and efficiency. While determining an operational methodology, the provided insights have the potential to improve the performance of decision support systems with almost 20%, in an effortless manner.

7 New benchmark instances

The results from Section 6 show that the performance of constructive heuristics suffers in case of a low multi-skilling level or short tasks. The exact solver performs worse when tasks are short, but also when the multi-skilling level is either high, or around 35%. Based on these observations, new benchmark instances were generated with short tasks ($\alpha \in \{120, 280\}$) and low multi-skilling level (20%, 30%).

Table 4 presents, for each new instance, the clique lower bound (LB), the best solution found by a general purpose solver after 1800 seconds (MIP),

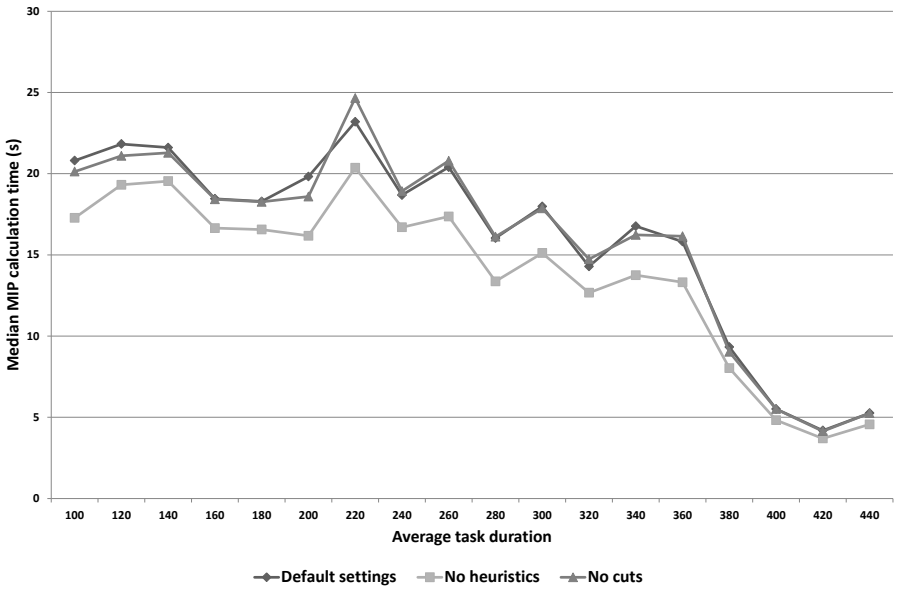

Figure 6: Median computation time in seconds for different configurations with varying average task duration (100 employees, 300 tasks, 60% skilling).

the best result found in ten repeated runs by CMH+LBIH ($b = 10$, $f = 4.5$) (F_{min}), the average result over ten runs (F_{avg}) and the average time required (T_{avg}).

Table 4 shows that a large part of the new instance set remains unsolved in the current experimental setting. A benchmark website[4] has been created to keep track of results on these instances.

8 Conclusions and future work

The present paper addresses the shift minimisation personnel task scheduling problem, which deals with the challenge of assigning tasks to employees who are restricted by qualifications and availabilities. The objective is to minimise the number of employees while still assigning all tasks.

A two-phase algorithmic design is proposed in which, building upon the matheuristic concept, both phases combine mixed integer programming with

[4] `http://allserv.kahosl.be/~pieter/smptsp.html`

			CMH+LBIH (b = 10, f = 4.5)		
Instance	**LB**	**MIP**	F_{min}	F_{avg}	T_{avg}
1_50_258_20	40	40	40	40.80	641.58
2_44_510_20	40	40	41	41.20	683.31
3_102_525_30	77	83	77	77.40	938.62
4_113_647_20	98	99	98	98.00	163.24
5_77_777_30	59	65	59	59.80	1615.46
6_135_777_20	116	119	116	116.90	1699.28
7_70_781_20	59	61	61	61.50	1800.00
8_88_1022_20	79	80	80	80.50	1800.00
9_125_1308_20	98	106	99	101.90	1800.00
10_153_1577_20	116	153	118	123.20	1800.00

Table 4: Computational results for the new benchmark dataset instances.

heuristic search, resulting in an efficient and versatile general optimisation method. Extensive computational experiments were performed to analyse relationships between different algorithmic parameters and the overall performance to justify the choices in design. The new hybrid algorithm finds, for the first time, optimal solutions for all 137 instances from a benchmark dataset, while consuming little calculation time. Experimental results demonstrate that, compared to previously previously published approaches, this novel algorithm holds the state of the art for the SMPTSP. In practice, the new algorithm can be used in an organisation's strategic planning phase to determine the optimal number of employees and staff skill mix. It is also fit for application as a decision support system to organise day-to-day operations such that an organisation's workforce can be more efficiently deployed. By providing an optimised resource occupation, the need for temporary workers is reduced or eliminated, thereby avoiding significant expenses for the organisation.

An empirical hardness study with three different constructive heuristics showed that the problem becomes harder when employees are low skilled, i.e. are qualified for relatively few tasks. Furthermore, the constructive heuristics showed to be sensitive to the average task duration. In problems with long tasks, these algorithms generated better solutions than when the tasks were shorter. The behaviour of a general purpose exact solver on the standard mathematical model was also studied. Experiments showed that the solver's performance deteriorates when the average task duration decreases, or when the staff is highly multi-skilled. These new insights into the com-

plexity of task scheduling problems can support changes of organisational policies, whereby the allocation of the available scarce resources can be better managed by e.g. adapting the staff skill mix to these findings.

A new benchmark dataset was generated consisting of instances satisfying the properties identified as hard. Many of these new, harder benchmark instances are still unsolved. We hope that researchers working on the SMPTSP will use these hard instances as an addition to the original dataset, and thus further challenge their algorithms' performance.

The empirical hardness study in the present paper observed the effect on algorithmic performance of the two most influential properties of the SMPTSP. Future research will focus on analysing and interpreting the results to better understand how certain problem features affect algorithms. In particular, the increase in computation time of the general purpose mathematical solver when employees can perform about one third of all tasks, presents an intriguing research question. Initial analysis suggests that the induced number of cliques is the underlying factor, however, a more elaborate interpretation is required to validate this claim. Furthermore, the new instances reveal the need for improved models and for algorithms that can better cope with hard problem features.

Acknowledgment: This research was carried out within the IWT 110257 project.

References

Bonomo, F., Duran, G., Marenco, J., 2006. Exploring the complexity boundary between coloring and list-coloring. Electronic Notes in Discrete Mathematics 25 (0), 41–47.

Burke, E., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H., 2004. The state of the art of nurse rostering. Journal of Scheduling 7 (6), 441–499.

Carter, M. W., Tovey, C. A., Jan. 1992. When is the classroom assignment problem hard? Operations Research 40 (S1), 28–39.

Della Croce, F., Grosso, A., Salassa, F., 2011. A matheuristic approach for the total completion time two-machines permutation flow shop problem.

In: Merz, P., Hao, J.-K. (Eds.), Evolutionary Computation in Combinatorial Optimization. Vol. 6622 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 38–47.

Della Croce, F., Salassa, F., 2012. A variable neighborhood search based matheuristic for nurse rostering problems. Annals of Operations Research, DOI: 10.1007/s10479-012-1235-x, 1–15.

Doerner, K., Schmid, V., 2010. Survey: matheuristics for rich vehicle routing problems. In: Blesa, M., Blum, C., Raidl, G., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics. Vol. 6373 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 206–221.

Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D., 2004. Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153 (1), 3–27.

Fischetti, M., Lodi, A., 2003. Local branching. Mathematical Programming Series B 98, 23–47.

Gupta, U., Lee, D., Leung, J.-T., 1979. An optimal solution for the channel-assignment problem. IEEE Transactions on Computers C-28 (11), 807–810.

Hanafi, S., Lazic, J., Mladenovic, N., Wilbaut, C., Crévits, I., 2010. New hybrid matheuristics for solving the multidimensional knapsack problem. In: Blesa, M., Blum, C., Raidl, G., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics. Vol. 6373 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 118–132.

Kolen, A., Lenstra, J., Papadimitriou, C., Spieksma, F., 2007. Interval scheduling : a survey. Naval Research Logistics 54 (5), 530–543.

Krishnamoorthy, M., Ernst, A., 2001. The personnel task scheduling problem. In: Yang, X., Teo, K., Caccetta, L. (Eds.), Optimisation methods and application. Kluwer, pp. 434–368.

Krishnamoorthy, M., Ernst, A., Baatar, D., 2012. Algorithms for large scale shift minimisation personnel task scheduling problems. European Journal of Operational Research 219, 34–48.

Leyton-Brown, K., Nudelman, E., Shoham, Y., 2002. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming. CP. Springer-Verlag, pp. 556–572.

Li, J., Burke, E. K., Curtois, T., Petrovic, S., Qu, R., 2012. The falling tide algorithm: A new multi-objective approach for complex workforce scheduling. Omega 40 (3), 283 – 293.

Maenhout, B., Vanhoucke, M., 2013a. An integrated nurse staffing and scheduling analysis for longer-term nursing staff allocation problems. Omega 41 (2), 485 – 499.

Maenhout, B., Vanhoucke, M., 2013b. Reconstructing nurse schedules: Computational insights in the problem size parameters. Omega 41 (5), 903 – 918.

Maniezzo, V., Stutzle, T., Voss, S. (Eds.), 2009. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. Vol. 10 of Annals of Information Systems. Springer.

Messelis, T., De Causmaecker, P., 2014. An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. European Journal of Operational Research 233 (3), 511–528.

Musliu, N., Schaerf, A., Slany, W., 2004. Local search for shift design. European Journal of Operational Research 153 (1), 51 – 64.

Ormeci, E. L., Salman, F. S., Yücel, E., 2014. Staff rostering in call centers providing employee transportation. Omega 43 (0), 41–53.

Smet, P., Vanden Berghe, G., 2012. A matheuristic approach to the shift minimisation personnel task scheduling problem. In: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling. PATAT. pp. 145–160.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L., 2013. Personnel scheduling: A literature review. European Journal of Operational Research 226 (3), 367 – 385.

Wright, P. D., Mahar, S., 2013. Centralized nurse scheduling to simultaneously improve schedule cost and nurse satisfaction. Omega 41 (6), 1042 – 1052.

Zeitlhofer, T., Wess, B., 2003. List-coloring of interval graphs with application to register assignment for heterogeneous register-set architectures. Signal Processing 83 (7), 1411 – 1425.